On the Gap Between Mathematical Modeling and Measurement Analysis for Performance Evaluation of the 802.15.4 MAC Protocol

Francois Despaux, Ye-Qiong Song, Abdelkader Lahmadi

On the Gap Between Mathematical Modeling and Measurement Analysis for Performance Evaluation of the 802.15.4 MAC Protocol

François Despaux
LORIA, University of Lorraine
Nancy, France
Email: francois.despaux@loria.fr

Ye-Qiong Song
LORIA, University of Lorraine
Nancy, France
Email: song@loria.fr

Abdelkader Lahmadi
LORIA, University of Lorraine
Nancy, France
Email: abdelkader.lahmadi@loria.fr

***Abstract*—IEEE 802.15.4 MAC protocol is the basis of many wireless sensor networks. Several studies have focused on analyzing the MAC layer by means of mathematical models such as Markov chain in order to be able to estimate the protocol performance parameters. Normally, simulation is used in order to validate the accuracy of the model. Unfortunately, real life is not always as easy as we would expect and extra considerations must be taken into account when considering real scenarios. In this paper our objective is to determine the existing gap between theoretical models and measurement analysis in real scenarios. We compare results obtained by an *a priori* accurate mathematical framework modeling the slotted IEEE 802.15.4 MAC protocol with experimental results obtained in Telosb motes and an implementation of the protocol over TinyOS. Comparision was made in terms of average delay. We also present some implementation considerations that needs to be taken into account when designing theoretical models for evaluating the delay in WSN.**

***Keywords*-TinyOS; Slotted CSMA/CA; Delay Evaluation**

I. Introduction

IEEE 802.15.4 protocol stack [1] is the basis of many wireless sensor networks (WSN) and has been proposed for low data rate and low power applications. Understanding the behavior and performance limitation of the protocol is challenging. In this way, a lot of research has been done to evaluate its performance through different methods, including theoretical modeling and simulation analysis. Bianchi's model [2] for the standard IEEE 802.11 under saturated traffic and ideal channel conditions have been extended for modeling the IEEE 802.15.4 MAC protocol. In [5], Park et al. proposes a Markov chain based on [2] for modeling the slotted version of the CSMA/CA mechanism and they give performance results in terms of service time delay and delay for successful packet sent. Misic et al. in [4] proposes a Markov chain approach for modeling the beacon enabled IEEE 802.15.4 MAC protocol considering M/G/1/K queue modeling and superframe with both active/only and active/inactive periods. Expressions for the access delay, probability distribution of the packet service time as well as probability distribution of the queue length are presented. Based on simulation results, both [5] and [4] frameworks seems to be suitable for estimating the delay in slotted CSMA/CA. However, none of the approaches consider the implication of implementing the protocol in real motes such as Telosb and how the underlying Operating System (OS), real hardware interaction and realistic conditions affects the delay estimation. In this paper we present an experimental evaluation of the Slotted CSMA/CA MAC protocol by considering realistic conditions using Telosb motes and an implementation of the protocol over TinyOS and we compare our results with the ones obtained in [4]. We present results in terms of average delay showing the gap between theoretical and empirical approaches and we give some implementation considerations that needs to be taken into account when designing theoretical models for evaluating the delay in WSN in order to have a more accurate model to work with. This paper is organized as follows. In Section II we present an overview of the IEEE 802.15.4 MAC protocol and the Markov chain model presented in [4]. A MAC layer implementation in TinyOS is presented in Section III. Experiments and Results are presented in Section IV and V respectively. A discussion regarding the results is given in Section VI and we conclude our work in Section VII.

II. IEEE 802.15.4 Standard & Markov Chain Model

A. Overview of IEEE 802.15.4 Standard

Misic The main idea of the MAC sub-layer in the IEEE 802.15.4 protocol is to provide an interface between the PHY layer and the higher layer protocols of LR-WPANs. Like the IEEE 802.11 protocol, the standard make use of CSMA/CA as the channel access protocol and it also brings support for contention-free and contention-based periods. Two operational modes are supported, *beacon enabled* and *Non beacon-enabled* modes. In the former beacons are periodically generated by the coordinator to synchronize attached devices. A beacon frame is part of a superframe which also embeds all data frames exchanged between the nodes and the PAN coordinator. In *Non beacon-enabled* devices can simply send their data by using unslotted CSMA/CA and there is no notion of superframe. In our work, we focus in the *beacon enabled* mode which use a slotted version of the CSMA/CA and exponential backoff. Both CSMA/CA

mechanism are based on backoff periods where one backoff period is equal to *aUnitBackoffPeriod*= 20 symbols (sym), $1\ sym = 4\ bits$ and the backoff period boundaries must be aligned with the superframe slot boundaries. Four variables are considered in the CSMA/CA mechanism:

NB: represents the number of times the CSMA/CA algorithm will enter in backoff while attempting the access to the current channel. It is initialized to zero before each new transmission attempt. If *NB* exceeds the limit of *macMaxCSMABackoffs*, the algorithm terminates with channel access failure status and failure is reported to higher protocol layers which can then decide whether to abort the packet in question or re-attempt to transmit it as a new packet.

RT: if a node fails to receive ACK due to collision or acknowledgement timeout the variable is increased by one up to *macMaxFrameRetries* and the packet is retransmitted. If RT is greater than *macMaxFrameRetries* packet is discarded due to the retry limit. This value is initialized to zero.

CW: representing the number of times the CSMA/CA will check the channel availability before starting transmission. The value by default is 2 and is reset to 2 each time the channel is busy and finally,

BE: represents the backoff exponent. Each time the channel is found busy BE is incremented by 1 until it reach the maximum possible value *aMaxBE* which is a constant defined.

The slotted CSMA/CA algorithm proceeds as follows.

- *1 - Initialization*: In this step, NB, CW, RT and BE are initialized. BE value is determined based on the *macBattLifExt* parameter. If its value is equal to *true* then BE is initialized as $BE = min(2, macMinBE)$, otherwise, it is initialized as $BE = macMinBE$ where *macMinBE* specifies the minimum of backoff exponent which is set to 3 by default. After the initialization, the algorithm locates the boundary of the next backoff period and goes to step 2.
- *2 - Random waiting delay for collision avoidance*: In this point the algorithm waits a random backoff in order to avoid collisions. The random backoff period is taken from the range of $[0, 2^{BE}-1]$. After the backoff period, it goes to step 3.
- *3 - Clear Channel Assessment*: In this step the algorithm check the availability of the channel. If the channel is found idle the algorithm goes the *idle channel* step, otherwise it moves to the *Busy channel* step. The CCA must be started at the boundary of a backoff period after the expiration of the waiting delay timer.
- *4 - Busy channel*: In this step, the CCA found the channel busy. Then, BE and NB parameters are incremented by 1 and CW is reseted to the initial value. The BE parameter must not exceed the value of *aMaxBE* parameter whose value by default is 5. If NB exceeds the parameter *macMaxCSMABackoffs* then the algorithms finish by throwing a channel access failure status. Otherwise, if NB is lower or equal to *macMaxCSMABackoffs* algorithm jumps to step 2 (*Random waiting delay for collision avoidance* step).
- *5 - Idle channel*: In this step, the channel was found idle. Then, CW parameter is decremented by 1. In case CW reaches zero then the MAC Protocol may start successfully its transmission. Otherwise, the algorithm returns to step 3. The transmission is started if and only if the remaining number of backoff periods in the current superframe is sufficient to handle both the frame and the subsequent acknowledgement transmissions. Otherwise, transmission is deferred until the next superframe. In case of a collision or acknowledgement timeout the algorithm goes to step 6. If packet is transmitted but a collision occurs, the algorithm continues in step 6.
- *6 - Collision Ocurred*: In this step RT is incremented by one. If RT is lower than *macMaxFrameRetries* then variables CW, NB and BE are initialized as in step 1 and algorithm jumps to step 2 (*Random waiting delay for collision avoidance* step). Otherwise, the packet is discarded due to the retry limit.

B. Markov Chain Approach

Authors in [4] present a Markov chain approach for modeling the behavior of the slotted IEEE 802.15.4 MAC protocol. In this approach, the packet queue in the device buffer is modeled as a *M/G/1/K* queueing system. Based on the protocol specification previously presented, authors model the system as a stochastic process $\mathcal{P}$=*{n(t), c(t), b(t), d(t)}* where $n(t)$ represents the value of the backoff time counter at time t, $c(t)$ represents the value of NB at time t, $b(t)$ represents the value of CW at time t and $d(t)$ represents the current value of the delay line counter (started if packet is deferred) at time t. Then, a discret-time Markov chain depicting this process is defined. Based on this Markov chain model, authors were able to find the probability distribution of the packet service time, probability distribution of the queue length and both probability distribution and average delay from the moment a packet arrives to the queue until the moment the node receives the corresponding packet acknowledgement. Our purpose is to compare this theoretical average delay with the empirical one. For details in the implementation refer to chapter 3 in [4].

III. MAC LAYER IMPLEMENTATION IN TINYOS

The objective of this section is to present an overview of the TKN154 MAC implementation over TinyOS in order to understand the main components of this module and also to determine how delay is affected by the interaction of each component.

A. TKN154 Module

TKN154 [3] is a platform independent IEEE 802.15.4-2006 MAC implementation for the 2.1 release of the TinyOS execution environment meeting the main tasks of the 802.15.4 MAC protocol such as PAN association and disassociation, slotted and unslotted versions of protocol, beacon transmission and synchronization, among others. Main components and interfaces used to exchange MAC frames between components are shown in Figure 1 and defined in [3]. TKN154 MAC can be divided into three sublayers. The lowest level, the RadioControlP component, manages the access to the radio. Components on the second level represent different parts of a superframe. For instance, BeaconTransmitP/BeaconSynchronizeP responsibles for transmission/reception of a beacon frame, DispatchSlottedCsmaP component manages frame transmission and reception during the CAP. The components on the top level implement the remaining MAC data and management services such as PAN association or requesting data from a coordinator. A component of this level typically provides MAC primitives to the next higher layer. For instance, DataP provides *MCPS-DATA.request* primitive to the next higher layer to send a frame to a peer device. Data frame will be assembled and enqueued in the send queue DispatchQueueP. DispatchSlottedCsmaP will eventually dequeue the frame and manage its transmission. Transmission status will be propagated to higher layer by means of *MCPS-DATA.confirm* event where an appropriate status code will signal whether the transmission was successful or not.
A set of interfaces towards the Radio Driver are also implemented. These interfaces push many time-critical operation from the MAC to the radio driver which are not negligible affecting the packet transmission delay.

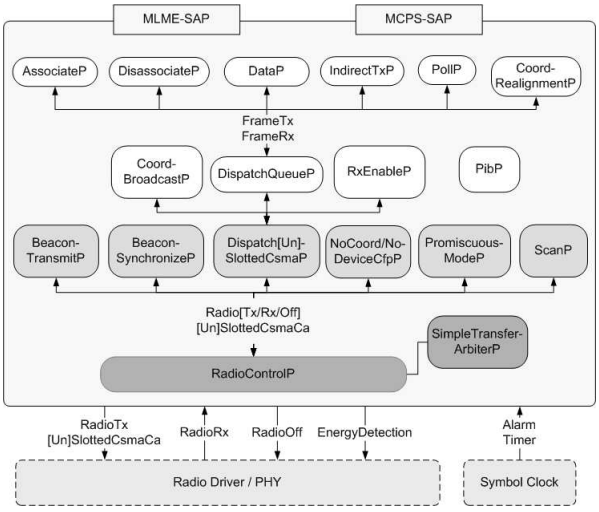


Figure 1: The TKN15.4 architecture: components are represented by rounded boxes, interfaces by connection lines.

B. Delay Analysis within TKN154 Implementation

As we mentioned before and based on simulation results, [4] framework approach seems to be accurate for estimating the delay within the slotted IEEE 802.15.4 MAC protocol. However, implementation considerations were not taking into account and as we will see next, these cannot be omitted if we want to have a suitable mathematical framework to estimates the delay within the protocol. In this subsection we analyse the TKN154 components interaction showing how delay is affected due to implementation issues. In TinyOS there are two threads of execution: tasks and hardware event handlers. Tasks are functions whose execution is deferred. Once scheduled, they run to completion and do not preempt one another. Hardware event handlers are executed in response to a hardware interrupt and also runs to completion, but may preempt the execution of a task or other hardware event handler. In this way, the completion of a particular task or event handler may be delayed due to a hardware interrupt affecting the delay of the whole protocol. Some other issues associated to the Operating System behavior may have effects on the protocol performance. For example, the TinyOS interface to the SPI bus introduces a large processing overhead that reduces the achievable SPI bus transfer rates [6].

1) Extra-delays estimation: In order to estimate extra-delays in the implementation of the MAC protocol over TinyOS we have analysed the execution stack from the moment a packet is generated until the moment the packet is acknowledged (or eventually lost if the number of retries exceeds the predefined threshold). Figure 2 shows a sequence diagram with the most relevant operations within the protocol. The first non-negligible delay presented in the execution is related to the SPI resource request in step 6. This delay together with the CC2420 switch to Rxmode shown in step 7 gives an extra delay of $t_1 = 5ms$. Then a random backoff period and two clear channel assessment (CCA) follows the execution of the protocol. However, both CCA and the random backoff period were taken into account in [4]. Next, a fix delay (step 11) of $t_2 = 2.8ms$ between the end of the second CCA and the invocation of the *transmissionStarted* function was found. Transmission is then delayed $t_3 = 2.8ms$ due to SFD Capture operation (step 13). Finally the packet is sent with a transmission delay shown in step 14. We have also found an extra-delay (step 15) $t_4 = 3ms$ in the coordinator side before sending the acknowledged packet back to the source (step 16). This analysis gives us un idea of the extra-delay due to the protocol implementation that should be taking into account when estimating the MAC protocol delay in TinyOS. Finally, we have also found a random delay t_5 due to deferred packets. Normally, when the remaining time within the CAP period of the current superframe does not suffice to complete the transmission the packet is deferred and

will be transmitted at the begining of the next superframe. However, in TinyOS we have seen that in some cases the deferred packet transmission does not start at the begining of the next superframe. Instead, it skips the immediate superframe deferring the transmission to the next one. That means that in this case, packet transmission will be delayed $t_{wait} = t_{curr} + t_{sup}$ where t_{curr} is the remaining time within the CAP period of the current superframe and t_{sup} is the duration of the superframe (in our case a 100% duty cycle is consider so optional inactive period is not taking into account). Considering that in our experimental scenarios, MAC attributes SO and BO were set to five then $t_{sup} = 30720$ symbols meaning that delay in those packets skiping the immediate superframe would be at least 30720 symbols = 0.5 seconds (1 symbol = 16μs).

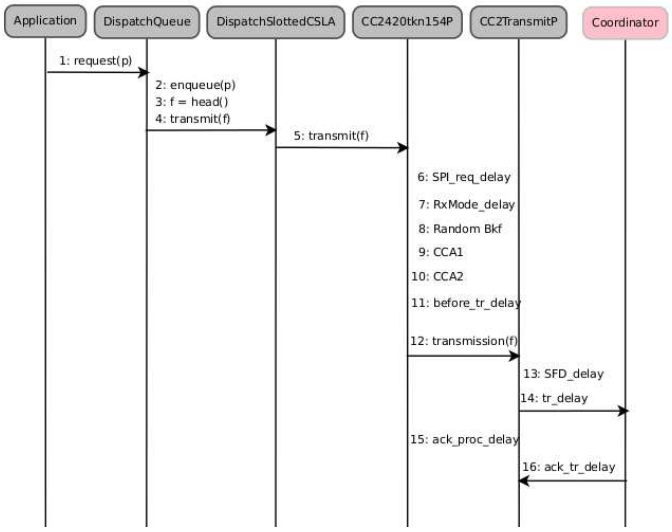


Figure 2: TKN154 Delay Analysis.

IV. Experiments

In this section we present the experiments we have done in order to compare the theoretical results obtained by [4] with real scenarios using an implementation of the slotted IEEE 802.15.4 MAC protocol over TinyOS and Telosb motes. We first start by specifying the main parameters and then we present a set of scenarios we have used in the experimentation.

A. Parameters & Scenarios

In order to compare both theoretical and experimental results we have set the same MAC protocol parameters which are shown in Table I. As we can see both BO and SO are set to 5 so no inactive period is considered and then Duty Cycle is 100%. We consider six scenarios which varies in the number of nodes and arrival rate of packet to nodes. The idea is to make a comparition of both approaches in terms of the average delay, that is to say, the average delay

Parameter	Value
Max Frame Retries	3
Max CSMA Backoff	4
Max Backoff Exponent	5
Min Backoff Exponent	3
Battery Life Extension	False
Beacon Order	5
Superframe Order	5

Table I: MAC parameters.

from the moment a generated packet is put in the queue until the reception of the acknowledged for that packet. We consider a star topology where coordinator is situated at the center and devices are located around the coordinator. Distance between devices and coordinator is the same for all scenarios and was set to one meter. The transmission power for each node was set to 0dBm. We know from [7] that, for this transmission power and considering a distance of one meter, the packet reception rate is almost 1 since the transitional region (a region characterized by unreliable and asymetric links with high variance in reception rate) starts at a distance of almost 10 meters. Simulation parameters are summarized in Table II. Packet payload is fixed as 34 bytes for all scenarios. As in [4], packet arrivals to each device follow a Poisson process with mean arrival rate of λ and each node accepts new packets through a buffer with finite size of $L = 2$ packets. All scenarios use the same buffer size. Channel bitrate is 250kbps.

Scenario	Nodes	Traffic Load λ (packets/s)
Scenario 1	2	1
Scenario 2	4	1
Scenario 3	6	1
Scenario 4	2	10
Scenario 5	4	10
Scenario 6	6	10

Table II: Scenario Parameters

V. Results

We present now the experimentation results. Our objective is to compare the theoretical and empirical average delay. Misic framework was implemented in Matlab while we use TKN154 implementation of the slotted IEEE 802.15.4 over TinyOS and telosb motes. For each scenario, a total of fifteen measurements were done in telosb motes and then the average delay was found and compared with the average delay obtained by the [4] framework. Table III shows theoretical and empirical average delay for each scenario. Figures 3 and 4 summarize the results of each scenario. Points in graphics represent the empirical measures (a total of fifteen instances of measurement for each scenario). Each empirical measure (point in graphics) is the result of the analysis of all the packets that were generated during the current instance of measurement. We took the average of these delays in order

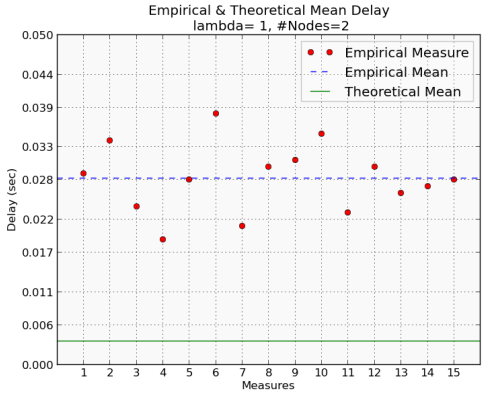
to determine the empirical measure of delay for the current instance and we plot the corresponding point. This procedure is repeated fifteen times for each scenario. Blue-dashed line represents the empirical average delay (average of fifteen points) for each scenario, while the green line shows the average delay obtained by [4] mathematical framework.

Scenario	Theoretical Av. Delay (sec)	Empirical Av. Delay (sec)
Scenario 1	0.00356	0.028
Scenario 2	0.0036	0.030
Scenario 3	0.0036	0.031
Scenario 4	0.0038	0.040
Scenario 5	0.0043	0.042
Scenario 6	0.005	0.044

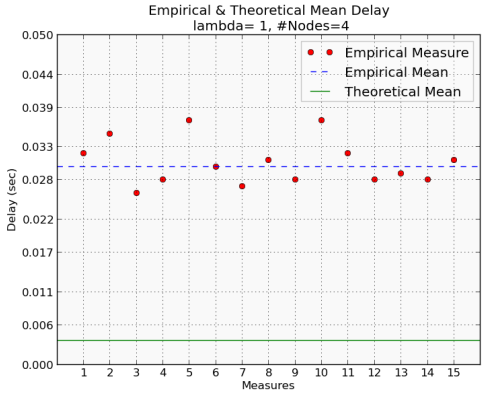
Table III: Theoretical VS Empirical Delay.

VI. Discussion

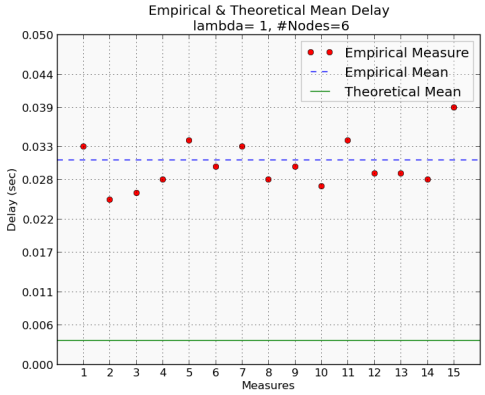
As we can see from Figure 3, 4 and Table III it is evident that there is an important gap between empirical and theoretical results. We see from results that when the number of nodes increases the delay also increases. This is normal since as the number of nodes grows, the number of times the channel is found busy also increments. This behavior is expected as well for the number of collisions. We also see that delay when considering a mean arrival rate of $\lambda = 10$ is greater than the one found for scenarios having $\lambda = 1$. This is due to the fact that for scenarios having mean arrival rate of $\lambda = 10$, the queue is expected to be busy during a non negligible time. That means that delay for those packets in the queue will increase since it would have to wait until the acknowledgement from the previous sent packet arrives. An important issue we found is regarding the first scenario. In this case we have two nodes and mean arrival rate $\lambda = 1$ so we do not expect to have too much collisions, channel is expected to be idle most of the time and in theory, the queue should be in idle state most of the time. However, the gap between theoretical and empirical delays is almost 24ms. How can we explain this behavior ?. By analysing the TinyOS traces during the experiment we found that a number of deferred packets skips the immediate superframe delaying the transmission to the next one. For these cases, the node's queue would not be necessary idle most of the time as in theory we would expect for this low-traffic scenario. This happens because deferred packets would have an extra-delay and then packets arriving would find the node busy and they will have to wait until the deferred packet is acknowledged. Then, contrarily to the expected behavior, the queue won't be idle most of the time and this fact will impact in the average packet delay. [4] Markov chain model is a very good approach for modeling the IEEE 802.15.4 MAC protocol considering and covering the main aspects of the protocol behavior (packet collisions, packet deferring, random backoffs, etc). Simulation results shown in [4] validate the mathematical approach. However, as we discussed in section



(a) Scenario 1.

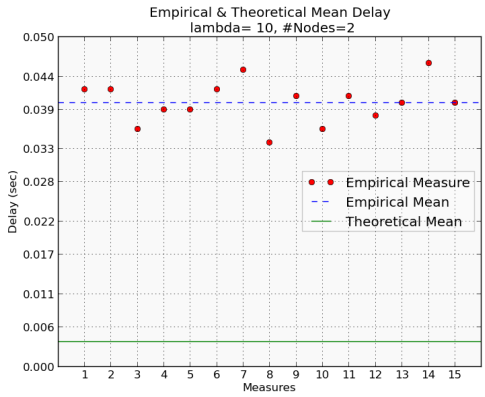


(b) Scenario 2.

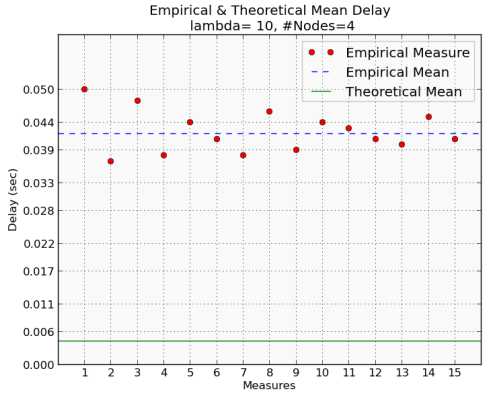


(c) Scenario 3.

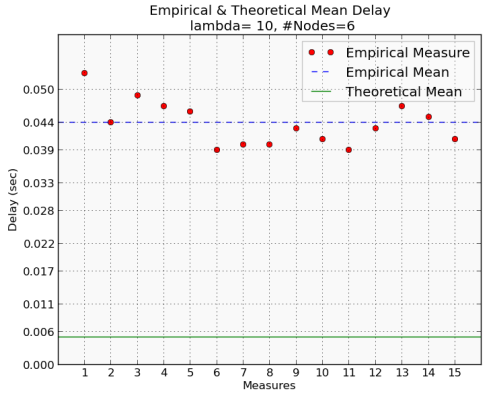
Figure 3: Traffic Load $\lambda = 1$ packets per seconds.



(a) Scenario 4.



(b) Scenario 5.



(c) Scenario 6.

Figure 4: Traffic Load $\lambda = 10$ packets per second.

II.B, non negligible delays arise when consider a realistic scenario with real motes due to operations of the underlying operating system. From the practical point of view, omitting the operating system features and behavior would leads to a useless model for analysing the protocol performance, in this case, the delay. A question that emerges immediately: is it possible to reduce the gap between theoretical and empirical results?. The first thing that we can do in order to reduce the gap is to focus our attention in the protocol implementation. In our analysis in section II.B, we detected an anomaly in the TKN154 implementation concerning the deferred packets. As we saw, some deferred packets skip the incoming superframe delaying the transmission until the reception of the next one (which amounts 0.5 seconds for the set of parameters defined here). So an important point to do is to detect anomalies in the implementation that could introduce extra-delays in the protocol and fix them. However, as shown in Figure 2, even considering an implementation exempt from anomalies and errors, delays are present due to function calls, hardware operations such as resources request, radio switch to reception/transmission mode, etc. Considering a free-error implementation, one approach then to reduce the gap is to quantify the existing delays in the implementation as done in section II.B and add them to the theoretical model in such a way that extra-delays are also considered when computing the average delay, for instance by changing (3.36) formule in [4].

VII. Conclusion

In this paper we have presented an analysis of the average delay in slotted IEEE 802.15.4 protocol in real scenarios considering the TKN154 implementation over TinyOS. Our objective was to determine the gap between a Markov chain approach for estimating the average delay presented in [4] and a real implementation in TinyOS. By analysing the execution of the protocol in real nodes we were able to determine constant and random delays inherent to the operating system operation such as hardware event handlers threw due to hardware interrupts whose execution can preempt a particular task and thus delaying its completion. Also we have noticed that some deferred packets skips the immediate superframe delaying the transmission to the next one. Even though we consider that [4] model is a good approach for estimating the average delay we conclude that it is incomplete since it does not consider the main aspects of real scenarios with an underlying operating system and real motes. One way to reduce this gap is to detect anomalies in the implementation side as the one we found in section II.B and fix them to have a free-error protocol implementation. Then, by considering this free-error implementation the next step should be to quantify the delays due to function calls, hardware operations, etc. and add them to the theoretical framework in order to have a realistic way for estimating packets delay within the protocol.

REFERENCES

[1] IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - specific requirement Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, 2007.

[2] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE J.Sel. A. Commun.*, 18(3):535–547, September 2006.

[3] Jan-Hinrich Hauer. Tkn15.4: An ieee 802.15.4 mac implementation for tinyos 2. TKN Technical Report Series TKN-08-003, Telecommunication Networks Group, Technical University Berlin, March 2009.

[4] Jelena Misic and Vojislav Misic. *Wireless Personal Area Networks: Performance, Interconnection, and Security with IEEE 802.15.4*. Wiley Publishing, 2008.

[5] P. Park, P. Di Marco, C. Fischione, and K. H. Johansson. Delay analysis of slotted IEEE 802.15. 4 with a finite retry limit and unsaturated traffic. In *IEEE Global Communications Conference*, page 18, 2009.

[6] Petcharat Suriyachai, Utz Roedig, and Andrew Scott. Implementation of a MAC protocol for QoS support in wireless sensor networks. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, page 16, 2009.

[7] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, page 517 526, 2004.